Evaluation of queue designs for true fully adaptive routers

Yungho Choia and Timothy Mark Pinkstonb,*

aIntel MMDC Corp., Shrewsbury, MA 01545, USA
bSMART Interconnects Group, EE-Systems Department, University of Southern California, Los Angeles, CA 90089-2562, USA

Received 27 April 2002; revised 21 July 2003

Abstract

In this paper, queue designs for true fully adaptive routers are explored. The proposed DAMQWR and VCDAMQ designs are aimed at exploiting the full capabilities of adaptive routing. Their effects on overall network cost and performance are evaluated analytically and by simulation. We show that DAMQWR best supports true fully adaptive routing capability by dynamically and efficiently managing queue and network resources, resulting in the highest network performance. Results show that up to 20% higher throughput can be obtained in comparison to traditional DAMQ designs.

© 2003 Published by Elsevier Inc.

Keywords: Interconnection networks; True fully adaptive router architecture; Queue designs

1. Introduction

Interconnection networks play a critical role in both multiprocessor systems (e.g., MPPs) and network-based computing systems (e.g., NOWs) by providing communication capability among processing nodes. With the recent emergence of multi-GHz CPUs and latency-sensitive, bandwidth-hungry applications including a variety of scientific simulations, multimedia servers, banking transactions, etc., interconnection networks may become a performance bottleneck in both MPPs and NOWs.

In order to accommodate these increasing network performance demands, many efforts have been made toward improving interconnection networks. One such improvement is the introduction of true fully adaptive routing algorithms [2,8,20]. The key benefit of true fully adaptive routing is to minimize message blocking in networks by eliminating routing restrictions enforced to prevent deadlock dependencies from possibly forming on resources. If deadlock behavior is detected, an efficient way of recovering from it is supported. Since these algorithms are implemented in routers, to fully exploit this capability, there is a need to optimize the router architecture [22,23].

In a router architecture, one of the major components associated with message blocking is the queue that temporarily stores messages until network resources become available. Therefore, to efficiently exploit the capability of the underlying routing algorithm, queue architectures must be optimized so that messages do not suffer unnecessary blocking and, thus, can be routed faster. In this regard, many queue architectures have been proposed such as first-in-first-out (FIFO), circular queue (CQ), fully connected circular buffer (FC-CB), statically allocated multi-queue (SAMQ), statically allocated fully connected multi-queue (SAFC), dynamically allocated multi-queue (DAMQ), dynamically allocated fully connected multi-queue (DAFC), centrally buffered dynamically allocated multi-queue (CBDA), high-performance input queue (HIPIQ), etc. [5,7,10–12,14,18,19]. However, these queue structures may not support true fully adaptive routing algorithms efficiently due to inherent head-of-line (HOL) blocking and/or fixed queue assignment limitations, as explained in Section 2.

This paper explores more optimal queue architectures for true fully adaptive routing algorithms. In doing this, this paper proposes two new queue architectures that can better exploit unrestricted routing capabilities of true fully adaptive routing algorithms: virtual channel...
DAMQs (VCDAMQs) and DAMQs with recruit registers (DAMQWRs). To evaluate and compare the performance of these new queue designs, routers incorporating the various queue designs are simulated at the network level. The next section describes various queue designs (including VCDAMQ and DAMQWR) and their unique features and limitations. Section 3 examines the cost of the presented queue designs and Section 4 provides extensive evaluation and performance analysis of various router queue designs including the proposed designs using network level simulation. Finally, the conclusions drawn from this work are summarized in Section 5.

2. Router queue designs

This section presents five queue designs and their usage in true fully adaptive router architectures: FIFO, CQ, DAMQ, DAMQWR, and VCDAMQ. The FIFO, CQ and DAMQ structures and variants of them are well-known queue designs that have been widely employed in commercial routers [6,12,13,16–18]. The DAMQWR and VCDAMQ structures are new queue designs proposed in this work intended for resolving critical limitations of CQs and DAMQs for true fully adaptive routing schemes.

2.1. FIFO and circular queues

As shown in Fig. 1, CQs are similar to FIFOs in that incoming packets are stored and scheduled in arrival order. However, unlike FIFOs, CQs do not require packets to traverse the entire length of the queue before being eligible for routing and scheduling. Instead, pointers demarcating the head and tail of the queue are used to insert and retrieve packets from the queue in FIFO order. The key benefit of CQs is reduced queuing latency as compared to FIFOs and design simplicity as compared to multi-queues. This reduces the architecture complexity of true fully adaptive routers as well as pin-to-pin latency, making this queue structure worthwhile to evaluate (i.e., we do not analyze FIFOs in the performance section).

The main problem of CQs and FIFOs, however, is head-of-line (HOL) blocking. HOL blocking is the situation in which a blocked packet at the head of the queue unnecessarily blocks subsequent packets following it that are heading for other—possibly free—output channels of a router. This can degrade network performance, making CQs and FIFOs suboptimal for true fully adaptive routers. A general solution to this problem is to multiplex multiple queues (virtual channels) over each physical channel [3,4]. This mitigates the HOL blocking problem by increasing the chance for packets to bypass blocked packets in other queues associated with the physical channel. However, this does not completely alleviate the problem as the number of virtual channels that can be implemented are limited. In addition, excessive virtual channels may complicate the router architecture by possibly increasing crossbar size and making routing and arbitration more complex [1,9]. This could, in effect, slow down the router clock cycle time and/or increase pin-to-pin latency.

2.2. Virtual output multi-queues

One input queuing technique that completely eliminates HOL blocking is to store packets into queues or subqueues of a multi-queue based on the output port to which they are to be routed. In essence, input queues act as virtual output queues, causing blocking only of those packets destined to the same output port. DAMQs, SAMQs and their variants (i.e., DAFCs, SAFCs, and CBDAs) fall into this class of queues. Here, we focus primarily on multi-queues that are dynamically allocated, although the results are also applicable to those that are statically allocated.

As shown in Fig. 2, a DAMQ is comprised of $K + 1$ subqueues in which space is dynamically maintained in the form of linked lists. There can be a DAMQ associated with each input port (as shown) or only one DAMQ associated with each router (i.e., a CBDA). In either case, each of the $K$ subqueues is associated with one of the $K$ output ports of the router for storing incoming packets assigned to those ports by the routing function. The remaining subqueue collects empty packet buffers, as shown. The multi-queue structure resolves the HOL blocking problem associated with CQs and FIFOs by eliminating routing interference between packets heading for different output links. Dynamically allocated multi-queues have the additional benefit of being able to efficiently adapt to network traffic by dynamically allocating queue space between the output ports based on network traffic demand.

Despite these performance merits, DAMQs suffer from a few complications and limitations. One problem has to do with the increased implementation complexity
caused by the linked lists and dynamic queue management. This complexity becomes more of a problem in DAFCs in which each subqueue has its own port to the router crossbar, as shown in Fig. 2(a). To reduce this complexity, DAMQs can provide only one reading port to the crossbar per input port, as shown in Fig. 2(b). This requires that the crossbar port be shared by all subqueues of a DAMQ, which could become a bottleneck and degrade performance.

Another major problem with DAMQs is that the queue structure is tailored more to deterministic routing algorithms than to true fully adaptive routing algorithms. In DAMQs, when a packet is incoming, a routing decision for the packet is made in order to assign the packet to one of the subqueues. This binds the packet to be routed only through the output link associated with the corresponding subqueue. That is, an output port cannot be assigned to a packet that is not associated with the subqueue into which the packet is stored, even if the routing function could supply that port in a subsequent routing iteration. In essence, a packet loses its routing adaptivity once it is assigned to a subqueue. Our preliminary investigations show this situation possibly occurring 10% of the time. While this may not be as detrimental to look-ahead or deterministic routing techniques, such a restrictive structure may not be very suitable for true fully adaptive routers. What is needed is a virtual output queue structure that allows packets to be granted all freed outputs allowed by the routing function even after they are temporarily stored in a subqueue, thus allowing packets to retain their ability to adaptively and flexibly route at all times.

As has been made evident, FIFOs, CQs and DAMQs have some limitations in supporting true fully adaptive routing capability. Therefore, there is a need to develop
queue designs that better support true fully adaptive routers by efficiently resolving HOL blocking and fixed queue assignment problems associated with current FIFO, CQ and DAMQ structures.

2.3. Enhanced dynamically allocated multi-queue

To overcome the problems of traditional queue designs for true fully adaptive routers, this work proposes two enhanced dynamically allocated multi-queues: dynamically allocated multi-queue with recruit registers (DAMQWR) and virtual channel dynamically allocated multi-queue (VCDAMQ).

As shown in Fig. 3, we propose an architecture in which each subqueue has $K - 2$ associated recruit registers, each corresponding to a different output port excluding the output port assigned to the subqueue and the opposite directional output port (we assume backtracking is not allowed). For instance, Fig. 3(a) represents a DAMQWR associated with the X+ input link in a 2-D torus router having five output links X+, X-, Y+, Y-, and PRC, where PRC is the output port for the processing node. There is no X- subqueue for this port as backtracking of packets is not permitted by the routing function. As shown, the X+ subqueue has three recruit registers, one associated with each of the Y+, Y-, or PRC ports. Each of these recruit registers points to a packet in the subqueue that can be routed to the output port corresponding to that register, as determined by the routing function. Namely, the Y+...
recruit register of the X+ subqueue points to a packet in the X+ subqueue that can be routed toward the Y+ direction.

In general, as shown in Fig. 3(b), recruit register $R_2$ in subqueue S-1 points to a packet that is destined to all subqueue allowed to be routed through output port 2. In the event that the S-2 (Y+) subqueue has no packets and, therefore, is making no requests, the packet in the S-1 (X+) subqueue can be routed to the S-2 (Y+) output port instead. Thus, the main function of our proposed recruit registers is to help an empty and idle subqueue to actively recruit a packet from the other subqueues and, thus, to allow packets in otherwise congested resources (blocked subqueues) to be recruited into less congested resources if they are available. By scanning the information provided in the recruit registers, the routing and arbitration logic can more optimally allocate output ports to queued packets. However, this comes at a cost of additional delay incurred in recruit register updates and packet recruit operations if initiated. This could be hidden if recruit register updates can be overlapped with routing and subtype assignment, but, if not, this is likely to be much less than the delay experienced by a packet that blocks without the possibility of using a non-congested resource alternative.

Another enhanced dynamically allocated multi-queue structure proposed in this work is the VCDAMQ. This queue combines the virtues of DAMQ and CQ/FIFO, i.e., dynamic queue management and full utilization of all routing choices given. The difference between the VCDAMQ and the traditional DAMQ shown in Fig. 2 is that the subqueues of the VCDAMQ are associated with router virtual channels while those of the DAMQ are associated with router output ports. Hence, like DAMQs, VCDAMQs can efficiently adapt to unbalanced traffic loads among virtual channels by dynamically allocating queue space to virtual channels. In essence, VCDAMQs can be viewed as a set of size-varying CQs or FIFOs assigned to a physical link. Therefore, each subqueue of a VCDAMQ is not coupled to any specific output link. This allows packets to be routed in any direction no matter which queue the packet is stored, better accommodating true fully adaptive routing capability. This advantage comes with the similar cost of dynamic queue space management as is in traditional DAMQs. In addition, like CQs and FIFOs, the VCDAMQ does not completely eliminate the HOL blocking problem. However, this problem might be better mitigated by VCDAMQs since more flexibility on virtual channel queue sizes is provided.

### 3. Cost of queues

To provide insight on the relative implementation cost of the proposed queue designs, this section presents an implementation of the four queues, i.e., CQ, DAMQ, DAMQWR, and VCDAMQ. To reflect the latest implementation technology trends (i.e., large die size and small transistor feature size), the following assumptions are made for queues and routers: virtual cut-through switching, large queue size capable of storing at least 8 packets, and input queueing.

As shown in Figs. 4–6, these queues commonly consist of packet cells, demultiplexers, and multiplexers. A packet cell is a circular queue that can store a whole packet. The multiplexer (MX) and the demultiplexer (DM) select one of the packet cells to read and write, respectively. The reason why each queue design comprises packet cells rather than phit cells (a smaller data unit that can be transferred within one clock cycle through a physical link) is to provide a reasonable reduction in the cost of the large queues assumed in this work. That is, when a large queue consists of phit cells, the size of MX and DM required must be large. For instance, consider a 128 phit-deep queue with a packet size of 8 phits. In case of a queue comprising phit cells, a 128-to-1 demultiplexer/multiplexer is required to select one out of the 128 phit cells, which makes queues considerably slow. In contrast, if a queue consists of packet cells, a 16-to-1 demultiplexer/multiplexer is sufficient for the queue. Of course, in this case, multiple hierarchical accesses must be required to read and write a phit from/to a queue. Namely, to write and read a phit, this hierarchical queue design first selects a packet cell then, subsequently, selects a phit cell to read and write within the pre-selected packet cell, which can increase the average access time of queues. However, this increased access time can be minimized by exploiting the queue access behavior of virtual cut-through switching where data flowing between routers are controlled on a packet basis instead of a phit basis. With this switching, once a packet cell is selected, many subsequent queue accesses for reads and writes are done within the selected packet cell, which makes for the common case of queue accesses. This reduces average queue access time almost to the average time to access a phit cell without changing packet cells. Furthermore, the control process to select a packet cell and the control process to select a phit cell can be overlapped instead of being serialized. This is because the time when the next packet cell is selected can be predicted, i.e., right after the last phit of a packet cell is read. Consequently, this hierarchical queue design can reasonably reduce the cost of large queues, resulting in fast queues.

The main implementation difference between CQs and DAMQs (including VCDAMQs and DAMQWRs) lies in the control of packet cell accesses. By design, CQs always access packet cells sequentially. Therefore, as shown in Fig. 4, its input and output controllers, i.e., DMC and MXC, can be realized by two log $N$-bit incremental counters, where $N$ is the number of packet...
cells in CQ. In contrast, packet cell accesses in DAMQs are controlled by linked lists. For these linked lists, each DAMQ in Figs. 5 and 6 requires \((2K + N) (\log N)\)-bit pointer registers, where \(K\) and \(N\) are the number of subqueues and the number of packet cells in a DAMQ, respectively. As shown, each of the \(N\) pointers, named packet cell link pointers (Pnt), is coupled to one of the \(N\) packet cells and down-links the coupled packet cell to another packet cell. A total of \(2K\) pointers are needed for the head (H) and the tail (T) pointers, respectively, of the \(K\) subqueues that point to the packet cells located in the head and the tail of the subqueues. For example, consider the linked list for Subqueue 1 given in Fig. 5. The head pointer of Subqueue 1, i.e., the left most box located in Queue Control containing “H,” points to Packet Cell 1, which means that Packet Cell 1 is at the head of Subqueue 1 and, thus, is to be read. The packet cell link pointer coupled with Packet Cell 1, i.e., Pnt-PC1, points to Packet Cell 2, which down-links Packet Cell 1 to Packet Cell 2 in the linked list of Subqueue 1. In this way, packet cells are linked and form the linked list for Subqueue 1, as shown in the figure.

Besides the pointers mentioned above, DAMQWR requires additional \(2 \times (K - 1) \times (K - 2) (\log N)\)-bit recruit registers as shown in Fig. 6. Each subqueue of the DAMQWR (except for the PRC subqueue) has \(K - 2\) recruit registers pointing to the packet cells that can be routed to the other \(K - 2\) output links (excluding the
output link associated with the subqueue and the opposite directional output link). Each recruit register in a subqueue accompanies one pointer providing the up-link of the packet cell pointed to by the recruit register in the linked list. This pointer is required when a packet cell is recruited so that the associated linked list is updated. These linked lists and recruit registers increase the cost of the DAMQ and DAMQWR. However, the increased cost is negligible compared to the total cost dedicated to packet cells in a queue. For instance, consider a 128 phit-deep DAMQ and DAMQWR with a packet size of 8 phits and a phit size of two bytes assuming five subqueues (e.g., 2-D torus). The queue space required for packet cells is 256 bytes in both DAMQ and DAMQWR while the queue space for pointers of the DAMQ and DAMQWR is only 13 bytes and 28 bytes, respectively (only about 10% of the queue space is dedicated to packet cells). This makes queue speed a more critical issue than queue cost (overhead space).

The speed of queues can be evaluated in terms of two main queue operations: read and write. Due to the hierarchical access structure of queues, these two operations are again classified into intra-packet cell (intra-PC) and inter-packet cell (inter-PC) operations. An intra-PC operation consists of reading and writing a phit from/to a queue without changing packet cells while an inter-PC operation does so with changing packet cells. Note that, in the queue designs presented here, the packet cell designs of all the queues are identical, i.e., circular queue. This makes the delays of all intra-PC operations for all the queues equivalent.

In contrast, the queue designs presented have different delays for inter-PC operations due to their different queue management. First, take into account the inter-packet cell operation of performing a write. To find and select an empty packet cell for a phit to be written into, CQ simply increases the value of the counter controlling the demultiplexer DMC by one, according to the sequential access policy of CQ. In contrast, DAMQs (including VCDAMQs and DAMQWRs) load a pointer value from the Empty Cell Queue into DMC to select an empty packet cell, as shown in Figs. 5 and 6. While loading the incoming packet into the selected packet cell, DAMQs update a linked list in order to assign the incoming packet to a subqueue. Updating a linked list to write requires DAMQs to modify two pointers, i.e., the tail pointer and the packet cell link pointer in order to add an incoming packet to the tail of the linked list. Besides these pointer updates, if the incoming packet has additional routing choices, DAMQWR updates two more pointers, i.e., the recruit register and the associated up-link pointer. Although DAMQs need to update multiple pointers, these updates can be done independently and simultaneously, which does not significantly increase queue delay. Moreover, note that the inter-PC write delay of DAMQ and DAMQWR includes the delay of the routing decision logic because DAMQ and DAMQWR make routing decisions before storing packets, which is not included in the inter-PC write delay of CQ and VCDAMQ. Namely, although the inter-PC write delay of DAMQ and DAMQWR is higher than that of CQ and VCDAMQ, considering overall routing delay, the inter-PC write delay increase by DAMQ and DAMQWR can be negligible.

Regarding the inter-PC operation of performing a read, CQ increments the value of the counter (M XC) by one, to control the packet cell multiplexer and, thus, to select a packet cell to read from. In contrast, DAMQ, VCDAMQ, and DAMQWR commonly modify the...
head pointer of the subqueue such that the old packet cell in the head of the subqueue is replaced by the next packet cell in the associated linked list. Besides the pointer update, when a subqueue is empty, DAMQWR scans recruit registers to choose a packet to recruit and then updates the associated linked lists to recruit the packet. This operation is complicated and expensive compared to the other operations, resulting in high latency. However, considering that (1) this recruit operation is executed only when a subqueue has nothing to send and (2) DAMQWR recruits only packets that are not at the head of subqueues and are waiting idle, it is not difficult to see that the increased latency of the recruit operation does not hamper the performance of DAMQWR.

To provide insight on the speed of the queue designs presented, Table 1 provides the delays of five queue operations for six queue designs. Based on the implementation technology of the WARRP router [21], these queue delays are estimated by identifying the critical path of each queue operation and adding up the delay components along the critical path. For this queue delay evaluation, a queue depth of 128 phits, a packet size of 8 phits, and a two-dimensional torus network are assumed. Additionally, to minimize the HOL blocking of CQs and VCDAMQs, these queues are assumed to have four virtual channels.

As shown in the table, the delay of each intra-PC operation is the same for all the queue designs because their packet cell designs are identical. The reason why the intra-PC operation of writes takes a longer time than that of reads is that, with the implementation technology of the WARRP router, a write into a register has a longer delay than a read from a register. Regarding inter-PC operations, DAMQWR, DAMQ, and VCDAMQ have up to 77%, 68%, and 59% longer delays than CQ, respectively. This indicates that the dynamic queue management of DAMQs is more expensive than the static queue management of CQs. Furthermore, the complicated recruit function makes DAMQWRs slowest. However, the increased delay of DAMQs, i.e., DAMQ, VCDAMQ, and DAMQWR, does not significantly hamper overall queue and network performance. This is because intra-PC operations are more common than inter-PC operations in virtual cut-through (VCT) switching. For instance, in VCT, to write an 8 phit packet to a queue, one inter-PC write operation for the first phit and 7 intra-PC write operations for the subsequent phits are required. As a result, average queue access delay is dominated by intra-PC operation latency. Furthermore, considering blocking (queueing) time in routers, the effect of the increased queue access time of DAMQ, DAMQWR, and VCDAMQ on network performance would be negligible—considered as a second-order effect. Therefore, this work assumes all the queue designs have an equivalent queue access time.

### 4. Network performance evaluation

This section evaluates and compares the performance of various queue designs using *FlexSimNA*, a more enhanced version of *FlexSim 1.2* [15] in which queues are non-atomically allocated, capable of storing physical units (phits) from different packets simultaneously. For this evaluation, a true fully adaptive router similar to the WARRP router presented in [21] is assumed, and four of the queue designs presented in Section 2 are compared: CQ, VCDAMQ, DAMQ, and DAMQWR. Full connection to the router crossbar is assumed for all queue designs (i.e., FC-CQ and DAFC). All simulations assume a virtual cut-through 16 × 16 bidirectional torus network, with packets consisting of 8 phits each. A queue depth of either 64 or 128 phits is assumed, representing the trending larger queue sizes of commercial multiprocessor routers (i.e., 8 and 16 packets deep). All router designs use one injection and one delivery processor node port. For the CQ and VCDAMQ designs, four virtual channels per physical channel are assumed. A true fully adaptive minimal routing scheme based on progressive deadlock recovery (*Disha-Sequen-\*tial* [20]) is assumed with a default time-out of 25 cycles before deadlock is suspected. Each simulation is run for a duration of 50,000 simulation cycles beyond the initial transient period (the first 10,000 cycles) so that data is collected during steady state only. Each router design is evaluated by measuring maximum network throughput (in phits/node/cycle) and network latency (in cycles).\(^1\) This gives insight into how well each queue design can accommodate true fully adaptive routing capability, pointing to the more optimal queue design for high performance network router architectures.

\(^1\)Results are plotted in Burton Normal Form wherein each successive point in each curve represents an increase in load rate (of 0.1 or 0.05).

<table>
<thead>
<tr>
<th></th>
<th>CQ(4VC) (ns)</th>
<th>DAMQ (ns)</th>
<th>VCDAMQ (ns)</th>
<th>DAMQWR (ns)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write</td>
<td>2.2</td>
<td>2.2</td>
<td>2.2</td>
<td>2.2</td>
</tr>
<tr>
<td>(Intra-PC)</td>
<td>2.2</td>
<td>3.7</td>
<td>3.5</td>
<td>3.9</td>
</tr>
<tr>
<td>Read</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
<td>2.0</td>
</tr>
<tr>
<td>(Intra-PC)</td>
<td>2.0</td>
<td>3.3</td>
<td>3.3</td>
<td>3.3</td>
</tr>
<tr>
<td>Recruit-Read</td>
<td>N/A</td>
<td>N/A</td>
<td>N/A</td>
<td>6.2</td>
</tr>
</tbody>
</table>
Uniform traffic results: To examine the performance of the queue designs, each router design is simulated by varying queue size from 64 to 128 phits under a random (uniform) traffic pattern. Results are presented in Fig. 7. For smaller queues (64 phits deep), CQ, VCDAMQ, and DAMQWR have the highest maximum throughput (15% higher maximum throughput than DAMQ). This indicates that CQ, VCDAMQ and DAMQWR better support the given routing adaptivity by allowing packets to be routed through any direction while the queue structure of DAMQ limits routing adaptivity by tying packets to a routing direction. For larger queues (128 phits deep), the maximum throughput of DAMQ and DAMQWR increases by up to 7% while that of CQ and VCDAMQ increase by less than 2%. This means that, DAMQ and DAMQWR benefit from larger queue size in order to fully utilize their advantages, i.e., no HOL blocking and dynamic queue allocation. Nevertheless, DAMQ still has the lowest maximum throughput and highest latency in both cases of 64 and 128 phit-deep queues (up to 17% lower maximum throughput than DAMQWR). This indicates that the DAMQ structure considerably limits network performance by severely limiting routing adaptivity.

It can also be observed that, regardless of queue size, the VCDAMQ has slightly lower performance than the CQ (1% lower throughput). This shows that VCDAMQ often makes traffic loads among virtual channels unbalanced by dynamically and not equally allocating queue space among virtual channel networks. Indeed, as shown in [16], network performance is maximized when loads among virtual channels is balanced. Violating this, VCDAMQ slightly reduces its maximum throughput as shown. In contrast, CQ statically but more evenly allocates queue space to virtual channels, which helps to balance loads among virtual channels and increases network throughput for the case of uniform traffic. Of all of the queue designs, the DAMQWR provides marginally higher performance—up to 4% higher maximum throughput than the other queues. This result indicates that DAMQWR best supports the true fully adaptive routing through its recruit capability while efficiently utilizing queue space.

Non-uniform traffic results: To evaluate the capability of the queue designs to handle hot spots and biased traffic efficiently, each is evaluated under a non-uniform traffic pattern: bit-reversal. Results are presented in Fig. 8. As shown, DAMQWR and VCDAMQ provide the highest performance—up to 7% higher maximum throughput than CQ. The dynamic queue allocation capability of DAMQWR and VCDAMQ helps to efficiently handle the biased and congested traffic generated by the bit-reversal pattern. It is also observed that DAMQ provides the lowest performance; DAMQWR provides 20% higher throughput than DAMQ. Moreover, the performance gap between DAMQ and DAMQWR is marginally wider for non-uniform traffic as compared to the uniform traffic case (20% vs. 17%). This indicates that, indeed, DAMQ does not well support true fully adaptive routing capability and suffers more in those situations in which routing adaptivity is most useful.

Overall, regardless of network traffic, the DAMQWR is shown to exploit true fully adaptive routing capability
5. Conclusion

This paper explores queue designs for optimizing true fully adaptive router architectures. Queue designs are evaluated by examining their unique features, limitations, and overall effect on network performance. In doing this, this paper proposes and characterizes two enhanced dynamically allocated multi-queue designs: dynamically allocated multi-queue with recruit registers (DAMQWRs) and virtual channel dynamically allocated multi-queues (VCDAMQs). These queue structures are designed to overcome the limitations of traditional queues for true fully adaptive routers.

Through simulation, this work shows that the proposed DAMQWR design best accommodates true fully adaptive routing capability. The recruit function of DAMQWR resolves the structural problem of DAMQs, i.e., limited routing adaptivity exploitation, while fully inheriting the benefits of DAMQs. This maximizes network resource utilization while efficiently adapting to dynamic network traffic.

In addition, simulation results show that the VCDAMQ design also exploits true fully adaptive routing capability while mitigating HOL blocking, thus efficiently handling hot spots in non-uniform traffic patterns. This results in the second best performance. In contrast, although virtual channels of CQs and FIFOs can also mitigate HOL blocking, their static queue management is shown not to deal well with hot spots and network congestion, leading to the third best performance. The poorest performing queue structure is the DAMQ, which is shown to significantly limit true fully adaptive routing capability due to fixed queue assignment. This queue structure is best used in look-ahead or deterministic routers which inherently restrict routing flexibility of packets.

References


Dr. Yungho Choi, Intel MMDC Corp

YUNGHO Choi completed his B.S. in electrical engineering from the Yonsei University, Korea, in 1991 and his M.S. and Ph.D in electrical engineering and computer engineering from the University of Southern California in 1995 and 2001, respectively. Currently, he is a senior hardware engineer at Intel in Shrewsbury, Massachusetts. His research interests include interconnection network router architecture, network interface protocol, and cache coherence protocol for high performance massively parallel processor and network of workstation systems.

Dr. Timothy M. Pinkston, Associate Professor, Computer Engineering Division of the Electrical Engineering-Systems Department, University of Southern California

Timothy Mark Pinkston completed his B.S.E.E. degree from The Ohio State University in 1985 and his M.S. and Ph.D. degrees in electrical engineering from Stanford University in 1986 and 1993, respectively. Prior to joining the University of Southern California (USC) in 1993, he was a Member of Technical Staff at Bell Laboratories, a Hughes Doctoral Fellow at Hughes Research Laboratory, and a visiting researcher at IBM T. J. Watson Research Laboratory. Presently, Dr. Pinkston is an Associate Professor in the Computer Engineering Division of the EE-Systems Department at USC and heads the SMART Interconnects Group. His current research interests include the development of deadlock-free adaptive routing techniques and optoelectronic network router architectures for achieving high-performance communication in parallel computer systems—massively parallel processor and cluster computing systems. Dr. Pinkston has authored over sixty refereed technical papers and has received numerous awards, including the Zumberge Fellow Award, the National Science Foundation Research Initiation Award, and the National Science Foundation Career Award. Dr. Pinkston is a member of the ACM and a Senior Member of the IEEE. He has also been a member of the Program Committee for several major conferences (ISCA, HPCC, ICCP, IPPS/IPDPS, ICDCS, SC, CS&I, CAC, PCRCW, OC, MPPOI, LEOs, WOCS, and WON), the Program Chair for HiPC’03, the Program Vice-chair for EuroPar’03 and ICPADS’04, the Program Co-chair for MPPOI’97, the Tutorials Chair for ISCA’04, the Workshops Chair for ICPP’01, and the Finance Chair for Cluster 2001. He recently concluded two 2-year terms as an Associate Editor for the IEEE Transactions on Parallel and Distributed Systems.